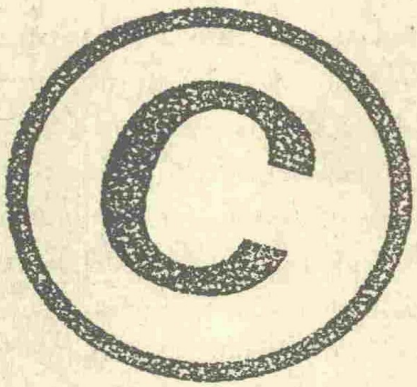
จเร เลิศสุดวิชัย

การใช้งานอินเตอร์รัพต์ใน Turbo C

ในเทอร์โบซีฉบับนี้ ทั้งเสริมความรู้ และให้ตัวอย่าง ในการพัฒนาโปรแกรม



ในฉบับนี้เป็นเรื่องของอินเตอร์รัพต์และฟังก์ชันบริการอินเตอร์รัพต์ในเทอร์โบซี ซึ่งจะมีตัวอย่างโปรแกรม และวิธีการสร้างโปรแกรมที่ใช้งานฟังก์ชันบริการอินเตอร์รัพต์มาให้ดูประกอบ รวมทั้งจะได้อธิบายถึงทฤษฎีของการอินเตอร์รัพต์ และโครงสร้างของอินเตอร์รัพต์ในระบบเครื่องไมโครคอมพิวเตอร์ที่สำคัญ ซึ่งเป็นส่วนที่ต้องถูกใช้งาน หัวข้อเรื่องที่จะกล่าวถึงในตอนนี้ได้แก่ เรื่อง

- ชนิดของการอินเตอร์รัพต์
- ตารางอินเตอร์รัพต์และการเชื่อมโยงเพื่อใช้งาน
- โปรแกรมเรสซิเดนต์อย่างง่าย ๆ

อินเตอร์รัพต์คืออะไร

การอินเตอร์รัพต์ก็คือ การขัดจังหวะการทำงานของโปรแกรมอย่างทันทีทันใด ในขณะที่โปรแกรมกำลังทำงานอยู่ ซึ่งเมื่อโปรแกรมถูกขัดจังหวะซีพียูของระบบจะหยุดการทำงานของโปรแกรมนั้นเอาไว้ แล้วกระโดดข้ามไปทำงานในโปรแกรมย่อยที่เตรียมไว้ เพื่อจัดการหรือบริการให้กับเหตุการณ์ที่ทำให้เกิดการอินเตอร์รัพต์ และเมื่อการทำงานในโปรแกรมย่อยสิ้นสุดลง ซีพียูของระบบก็จะกลับมาทำงานในโปรแกรมที่ถูกขัดจังหวะในตำแหน่งเดิม ในเครื่องระดับไมโครคอมพิวเตอร์การอินเตอร์รัพต์แบ่งออกเป็น 2 ลักษณะคือ

- ฮาร์ดแวร์อินเตอร์รัพต์ เป็นการอินเตอร์รัพต์จากอุปกรณ์ที่เชื่อมกับเครื่องขอบริการต่อซีพียู เช่น การอินเตอร์รัพต์จากแป้นพิมพ์ การอินเตอร์รัพต์จากนาฬิกาภายในเครื่อง หรือการอินเตอร์รัพต์จากช่องสื่อสารอนุกรม ฯลฯ
- ซอฟต์แวร์อินเตอร์รัพต์ เป็นการอินเตอร์ที่เกิดขึ้น เมื่อซีพียูได้รับคำสั่ง INT ซึ่งส่วนใหญ่มักจะเป็นโปรแกรมที่ต้องการใช้การบริการโดยตรงจากไบออส ดอส ดีไวซ์ไดรเวอร์ หรือโปรแกรมเรสซิเดนต์บางโปรแกรม

ชุดโปรแกรมย่อยที่ซีพียูกระโดดไปทำงานเมื่อเกิดการอินเตอร์รัพต์ขึ้น เรียกว่า โปรแกรมบริการอินเตอร์รัพต์ (Interrupt Service Routine) ซึ่งโดยส่วนใหญ่แล้วจะมีตำแหน่งที่อยู่ประจำ ระบบของการอินเตอร์รัพต์นั้นมีข้อดีคือ โปรแกรมสามารถตอบสนองต่อเหตุการณ์ภายนอกได้โดยที่ไม่ต้องคอยเฝ้าตรวจสอบตลอดเวลา (polling) ซึ่งในระบบ

ที่ใช้วิธีการเฝ้าตรวจสอบการทำงาน โปรแกรมหลักจะต้องคอยทดสอบเหตุการณ์ภายนอกตลอดเวลา ตรงกันข้ามกับระบบการอินเตอร์รัพต์ที่เหตุการณ์ภายนอกเป็นตัวไปกระตุ้นให้โปรแกรมที่ทำงานอยู่ทราบเอง แต่ในบางระบบที่ข้อจำกัดเป็นเรื่องของเวลา และจำนวนเหตุการณ์ที่ต้องตอบสนองก็มักจะใช้ระบบทั้งสองผสมกัน เช่นจากเหตุการณ์ภายนอกที่เข้าไปกระตุ้นโปรแกรมก็จะถูกจัดเก็บลงในคิว (queue) ที่เตรียมไว้ และโปรแกรมจะคอยดึงเอาเหตุการณ์ที่ร้องขออินเตอร์รัพต์จากตัวดังกล่าวมาพิจารณา แล้วให้บริการตามที่ต้องการ

ในบางครั้งอินเตอร์รัพต์ที่เกิดขึ้นอาจจะไม่เกี่ยวข้องโดยตรงกับโปรแกรมที่ทำงานอยู่ขณะนั้น และก็ไม่มีผลกระทบต่อการทำงานของโปรแกรม ตัวอย่างเช่น อินเตอร์รัพต์ที่เกิดจากฐานเวลาของเครื่อง (Timer) ซึ่งจะเกิดการอินเตอร์รัพต์ 18.2 ครั้งในช่วงเวลา 1 วินาทีและการอินเตอร์รัพต์ดังกล่าวมีงานที่จะต้องทำคือการนับนาฬิกาของระบบให้เดินตามปกติ ซึ่งจะเห็นว่าอินเตอร์รัพต์จากเวลาดังกล่าวไม่มีผลกระทบใด ๆ ต่อการทำงานของโปรแกรม แต่อาจจะมีผลทางอ้อมได้เช่น โปรแกรมที่สร้างไฟล์โดยผ่านทางดอสในรายละเอียดของไฟล์จะมีการบันทึกเวลาที่ทำการสร้างเอาไว้ด้วย ซึ่งเวลาดังกล่าวจะได้มาจากนาฬิกาของระบบ และบ่อยครั้งเหมือนกันที่การอินเตอร์รัพต์ต้องมีความสัมพันธ์กับโปรแกรมที่ทำงาน หรือโปรแกรมดังกล่าวต้องใช้งานอินเตอร์นั้นโดยตรงเลย ตัวอย่างเช่น โปรแกรมที่ต้องทำงานในช่วงเวลาที่แน่นอน ซึ่งต้องอาศัยระบบเวลาของเครื่อง ลักษณะนี้จะพบว่าการอินเตอร์รัพต์ของเวลาต้องเป็นส่วนหนึ่งของโปรแกรมด้วย หรืออีกตัวอย่างหนึ่ง คือ การอ่านข้อมูลจากพอร์ตสื่อสาร การอินเตอร์รัพต์ของพอร์ตสื่อสารก็จะเป็นส่วนที่เกี่ยวข้องอย่างใกล้ชิดกับโปรแกรม

เนื่องจากการอินเตอร์รัพต์จะหยุดโปรแกรมที่กำลังดำเนินการอยู่ขณะนั้นเพื่อไปทำงานในโปรแกรมย่อย เพื่อบริการเหตุการณ์ที่เป็นสาเหตุการอินเตอร์รัพต์ ดังนั้นจึงจำเป็นที่โปรแกรมบริการอินเตอร์รัพต์จะต้องเก็บสภาวะต่าง ๆ ของระบบเอาไว้เพื่อที่จะคืนสภาวะเดิมได้อย่างถูกต้อง ทุกครั้งที่เกิดการอินเตอร์ก่อนจะกระโดดข้ามไปทำงานในโปรแกรมบริการ ซีพียูจะทำการเก็บเอาตำแหน่งโปรแกรมขณะนั้น (return address) พร้อมทั้งรีจิสเตอร์แฟล็กเอาไว้ในสแต็ก ดังนั้นจะเห็นว่าสิ่งที่ในโปรแกรมบริการจะต้องทำเพิ่มก็คือการเก็บค่าต่าง ๆ ของรีจิสเตอร์ที่เหลือลงในสแต็กก่อนที่จะใช้งานมัน และจะต้องคืนค่าเหล่านั้นกลับออกมาด้วยก่อนจบการทำงาน และอีกจุดหนึ่งที่สำคัญคือคำสั่งรีเทอร์น ซึ่งเป็นคำสั่งสุดท้ายในโปรแกรมบริการจะต้องเป็นคำสั่ง IRET ไม่ใช่คำสั่ง RET ธรรมดาซึ่งในคำสั่งนอกจากการกลับคืนยังตำแหน่งโปรแกรมเดิมแล้วยังจะดึงค่ารีจิสเตอร์แฟล็กคืนจากสแต็กให้ด้วย

อินเตอร์รัพต์ทั้งสองประเภทคือฮาร์ดแวร์อินเตอร์รัพต์และซอฟต์แวร์อินเตอร์รัพต์ มีข้อแตกต่างกันอยู่บ้างอินเตอร์รัพต์เวลาซึ่งเป็นฮาร์ดแวร์อินเตอร์รัพต์อันหนึ่งทุกครั้งที่มีการกระตุ้นจากเวลาจะเกิดการขัดจังหวะขึ้น และโปรแกรมบริการที่เตรียมไว้สำหรับระบบเวลาก็จะถูกเรียกทำงาน อีกตัวอย่างหนึ่งคือ การอินเตอร์รัพต์ที่เกิดจากแป้นพิมพ์ทุกครั้งที่มีการกดแป้นพิมพ์จะเกิดการอินเตอร์รัพต์บอกซีพียูของเครื่อง ซึ่งเช่นเดียวกับการกระตุ้นจากเวลา โปรแกรมบริการอินเตอร์รัพต์ของแป้นพิมพ์จะถูกเรียกทำงาน ซึ่งเห็นได้ว่าในระบบยังมีอุปกรณ์พ่วงต่ออยู่อีกมาก ทุกครั้งที่อุปกรณ์หรือฮาร์ดแวร์เหล่านี้ทำให้เกิดการอินเตอร์รัพต์ ซีพียูจะต้องทราบว่าอุปกรณ์ใดเป็นผู้ร้องขอบริการและจะต้องใช้โปรแกรมบริการโปรแกรมใด ซึ่งปัญหาทั้งหมดจะถูกจัดการโดยตัวควบคุมอินเตอร์รัพต์และตารางอินเตอร์รัพต์เวกเตอร์ ที่เป็นตารางเก็บตำแหน่งโปรแกรมบริการต่าง ๆ

สำหรับซอฟต์แวร์อินเตอร์รัพต์ซึ่งส่วนใหญ่จะเกิดขึ้นเมื่อใช้คำสั่ง INT ของซีพียู ในโปรแกรมที่ต้องการเรียกใช้อินเตอร์รัพต์ ซึ่งถ้าพิจารณาให้ดีแล้วซอฟต์แวร์อินเตอร์รัพต์จะเหมือนกับโปรแกรมย่อยทั่วไปมากกว่าเพียงแต่ว่าวิธีการเรียกโปรแกรมย่อยนี้ไม่จำเป็นต้องรู้ตำแหน่งที่อยู่ของโปรแกรม เพราะการถูกเรียกใช้งานจะถูกเรียกผ่านจากตำแหน่งโปรแกรมที่เก็บไว้ในตารางอินเตอร์รัพต์เวกเตอร์ เหมือนกรณีของฮาร์ดแวร์อินเตอร์รัพต์ซึ่งซอฟต์แวร์อินเตอร์รัพต์ทั้งหมดจะได้แก่ โปรแกรมบริการของไบออสและดอสทั้งหมด ฉะนั้นจะเห็นว่าจากลักษณะที่คล้ายคลึงกับโปรแกรมย่อยทั่วไป ในตัวโปรแกรมบริการจึงมีการส่งค่าคืนกลับไปยังโปรแกรมที่เรียกใช้งาน โดยส่งค่าผ่านทางรีจิสเตอร์ต่าง ๆ ซึ่งไม่จำเป็นที่โปรแกรมบริการจะต้องรักษา หรือเก็บค่ารีจิสเตอร์เอาไว้เมื่อโปรแกรมบริการทำงานเสร็จเหมือนในฮาร์ดแวร์อินเตอร์รัพต์ แต่อย่างไรก็ตามฮาร์ดแวร์อินเตอร์รัพต์เองก็สามารถจะถูกเรียกได้ด้วยคำสั่ง INT เหมือนซอฟต์แวร์อินเตอร์รัพต์เช่นกัน

ตาราง อินเตอร์รัพต์เวกเตอร์

จากที่กล่าวไปแล้วในหัวข้อก่อน เมื่อเกิดอินเตอร์รัพต์ใดขึ้นมา ซีพียูจะกระโดดไปยังโปรแกรมย่อยของโปรแกรมที่ใช้บริการอินเตอร์นั้น ในการออกแบบระบบไมโครคอมพิวเตอร์จึงได้เตรียมตำแหน่งแอดเดรส (หรือที่อยู่) ของโปรแกรมบริการเอาไว้ขนาด 1 กิโลไบต์ ตั้งแต่แอดเดรส 0000:0000 ถึแอดเดรส 0000:03FF ซึ่งเรียกพื้นที่หน่วยความจำนี้ว่า "ตารางอินเตอร์รัพต์เวกเตอร์ (Interupt vector table)" ข้อมูลในตารางอินเตอร์รัพต์เวกเตอร์ประกอบด้วย แอดเดรสของโปรแกรมบริการของแต่ละอินเตอร์รัพต์ โดยที่แต่ละแอดเดรสจะใช้พื้นที่ข้อมูลขนาด 4 ไบต์ กำหนดให้ 2 ไบต์แรกเป็นออฟเซตและ 2 ไบต์ต่อมาเป็นเซกเมนต์ ดังในรูปที่ 1 ซึ่งพื้นที่ทั้งหมดจะสามารถบรรจุตำแหน่งของโปรแกรมย่อยบริการได้ทั้งหมด 256 อินเตอร์รัพต์ และในพื้นที่ส่วนนี้จะบรรจุเอาแอดเดรสของโปรแกรมและฟังก์ชันบริการของไบออสและดอส รวมทั้งแอดเดรสของโปรแกรมบริการฮาร์ดแวร์อินเตอร์รัพต์ เช่น เวลา เป็นพิมพ์ พอร์ตสื่อสารอนุกรม และบางส่วนก็เป็นที่วางสำหรับโปรแกรมอื่น ๆ จะใช้งานบ้าง

การหาตำแหน่งแอดเดรสของโปรแกรมบริการในตารางอินเตอร์รัพต์เวกเตอร์ของฮาร์ดแวร์อินเตอร์รัพต์จะต่างจากซอฟต์แวร์อินเตอร์รัพต์ อุปกรณ์ต่าง ๆ ที่ร้องขอบริการมายังซีพียูซึ่งจะต้องผ่านตัวจัดการอินเตอร์รัพต์ก่อนคือ ไอซี 8259 (ในกรณีเครื่อง AT จะมี 8259 สองตัว) ชิพ 8259 จะเป็นตัวคอยจัดระดับความสำคัญของอินเตอร์รัพต์ที่เกิดขึ้นแล้วกำหนดหมายเลขอินเตอร์รัพต์ส่งให้กับพีซียู เพื่อนำไปเรียกโปรแกรมบริการจากตารางอินเตอร์รัพต์เวกเตอร์อีกครั้ง ส่วนซอฟต์แวร์อินเตอร์รัพต์ซึ่งเกิดจากคำสั่ง INT แล้วตามด้วยหมายเลขขนาด 8 บิต หมายเลขดังกล่าวก็คือ หมายเลขอินเตอร์รัพต์ซึ่งตำแหน่งของโปรแกรมบริการจะหาได้จากหมายเลขดังกล่าวในตารางอินเตอร์รัพต์เวกเตอร์ ดังนั้นจะเห็นว่าทุกครั้งที่เกิดการอินเตอร์รัพต์ขึ้น ซีพียูจะทราบถึงหมายเลขของการอินเตอร์รัพต์ที่เกิดขึ้นจากนั้นจึงทำการเก็บค่าในรีจิสเตอร์แฟล็ก โค้ดเซกเมนต์ (Code segment) และรีจิสเตอร์ชี้ตำแหน่งคำสั่ง (Instruction pointer) ลงบนสแต็กแล้วจึงกระโดดไปทำงานยังโปรแกรมบริการตามหมายเลขอินเตอร์รัพต์ในตารางอินเตอร์รัพต์เวกเตอร์

การใช้งานอินเตอร์รัพต์

หลังจากได้ทราบทฤษฎีการทำงานของอินเตอร์รัพต์มาแล้ว ต่อไปก็จะเป็นเรื่องของการที่จะนำเอาอินเตอร์รัพต์ไปใช้งานจริง จะเห็นได้ว่าในระบบของไมโครคอมพิวเตอร์นั้นมีอินเตอร์รัพต์หลายอันที่ระบบต้องใช้งานอยู่เสมอ สำหรับการใช้งานอินเตอร์รัพต์ที่ว่างอยู่ก็คงจะไม่มีปัญหาอะไร แต่สำหรับอินเตอร์รัพต์ที่ถูกใช้งานโดยระบบดีไวซ์ไดรเวอร์หรือโปรแกรมเรสซิเดนต์ การใช้งานก็จะต้องมีวิธีการใช้เรียกที่ถูกหลัก

```
C:\>debug
-d0:0 00ff
0000:0000  8D 58 93 02 8E 07 70 00-16 00 51 0F 8E 07 70 00   .X....p...Q...p.
0000:0010  8E 07 70 00 54 FF 00 F0-93 9E 00 F0 93 9E 00 F0   ..p.T...........
0000:0020  AB 00 51 0F 25 01 51 0F-1A 02 51 0F A2 02 51 0F   ..Q.%.Q...Q...Q.
0000:0030  2A 03 51 0F B2 03 51 0F-3A 04 51 0F 8E 07 70 00   *.Q...Q.:.Q...p.
0000:0040  F6 0F 00 C0 4D F8 00 F0-41 F8 00 F0 39 10 3D 11   ....M...A...9.=.
0000:0050  39 E7 00 F0 59 F8 00 F0-CD 12 3D 11 D2 EF 00 F0   9...Y.....=.....
0000:0060  B2 58 00 F0 7C 1A 70 00-6E FE 00 F0 88 07 70 00   .X..|.p.n.....p.
0000:0070  53 FF 00 F0 A4 F0 00 F0-22 05 00 00 0A 46 00 C0   S........"....F..
0000:0080  8C 13 3D 11 F7 15 3D 11-F4 02 B0 17 2F 03 B0 17   ..=...=...../...
0000:0090  BC 02 B0 17 94 19 70 00-B9 19 70 00 87 18 3D 11   ......p...p...=.
0000:00A0  45 14 93 02 D1 06 70 00-45 14 93 02 45 14 93 02   E.....p.E...E...
0000:00B0  45 14 93 02 45 14 93 02-81 02 1F 10 76 24 3D 11   E...E.......v$=.
0000:00C0  EA 46 14 93 02 14 93 02-45 14 93 02 45 14 93 02   .F......E...E...
0000:00D0  45 14 93 02 45 14 93 02-45 14 93 02 45 14 93 02   E...E...E...E...
0000:00E0  45 14 93 02 45 14 93 02-45 14 93 02 45 14 93 02   E...E...E...E...
0000:00F0  45 14 93 02 45 14 93 02-45 14 93 02 45 14 93 02   E...E...E...E...
-
```

รูปที่ 1 แสดงค่าในตารางอินเตอร์รัพต์เวกเตอร์ที่ตำแหน่งหน่วยความจำตั้งแต่ตำแหน่ง 0000:0000 จำนวน 1 กิโลไบต์ ซึ่งมีด้วยกัน 256 อินเตอร์รัพต์ หรือ 256 ช่อง แต่ละช่องใช้ 4 ไบต์ โดย 2 ไบต์แรกเป็นเซกเมนต์และมี 2 ไบต์ถัดมาเป็นออฟเซตของโปรแกรมบริการ

เพื่อที่ทั้งโปรแกรมของเราและโปรแกรมเรสซิเดนต์ ดีไวซ์ไดรเวอร์ รวมทั้งระบบเครื่องทั้งหมดยังทำงานได้ตามปกติ จากหน้าที่ของตารางอินเตอร์รัพต์เวกเตอร์ในหัวข้อที่แล้ว จะพบว่าวิธีการที่เราจะใช้งานอินเตอร์รัพต์หมายเลขใดก็ทำได้โดยการเปลี่ยนค่าตำแหน่งโปรแกรมในตารางอินเตอร์รัพต์หมายเลขนั้น ให้เป็นค่าตำแหน่งหน่วยความจำของโปรแกรมย่อยที่เราเตรียมเอาไว้สำหรับบริการเมื่อเกิดการอินเตอร์รัพต์ขึ้น ตัวอย่างเช่น วิธีการสร้างระบบช่วยเหลือ (Help) ซึ่งตามปกติแล้วมักจะถูกเรียกขึ้นมาทำงานเมื่อผู้ใช้โปรแกรมกดแป้นพิมพ์ F1 ในการเขียนโปรแกรมแบบธรรมดานั้นก็มักจะต้องเพิ่มการตรวจสอบแป้นพิมพ์ F1 ทุกจุดของโปรแกรมที่มีการอ่านแป้นพิมพ์ หรืออาจจะต้องดัดแปลงฟังก์ชันการอ่านค่าแป้นพิมพ์ในไลบรารีให้มีการตรวจสอบ F1 แล้วให้เรียกระบบช่วยเหลือขึ้นมาทำงาน ซึ่งวิธีนี้ไม่เป็นที่นิยมใช้กัน (ในลักษณะนี้สามารถดัดแปลงโปรแกรมบริการของแป้นพิมพ์ได้) ซึ่งในโปรแกรมทั่วไป หรือดอสการอ่านแป้นพิมพ์ว่ามีแป้นใดถูกกดมักจะต้องเรียกผ่านอินเตอร์รัพต์หมายเลข 0X16 และค่าในตารางอินเตอร์รัพต์ช่องที่ 0X16 จะเก็บค่าตำแหน่งโปรแกรมย่อยในไบออสซึ่งเตรียมเอาไว้สำหรับบริการเมื่อมีใครต้องการทราบถึงการกดแป้นพิมพ์นั่นเอง แต่ถ้าเราเปลี่ยนแปลงค่าในตารางอินเตอร์รัพต์ในช่องที่ให้มีค่าตำแหน่งโปรแกรมย่อยของเรา (ระบบช่วยเหลือ) โดยในโปรแกรมย่อยจะตรวจสอบก่อนว่าแป้นพิมพ์ที่ถูกกดคือ F1 หรือไม่ ถ้าใช่ F1 ก็จะเรียกเอาระบบช่วยเหลือขึ้นมาทำงาน อย่างไรก็ตามการทำงานภายในโปรแกรมย่อยที่เราใส่เข้าไปแทนในระบบเดิมจะต้องเป็นไปตามกฎของโปรแกรมบริการอินเตอร์รัพต์เพื่อให้ระบบทำงานได้อย่างถูกต้องเหมือนเดิม

โปรแกรมบริการอินเตอร์รัพต์

จากตอนที่แล้วเราได้กล่าวถึงฟังก์ชันพิเศษในเทอร์โบซีที่เตรียมเอาไว้สำหรับทำหน้าที่เป็นโปรแกรมบริการอินเตอร์รัพต์ ซึ่งภายในฟังก์ชันดังกล่าวก่อนที่จะทำงานใด ๆ ก็จะมีการเก็บค่ารีจิสเตอร์ทั้งหมดลงสแต็กให้ก่อนและเมื่อเสร็จสิ้นโปรแกรมก็จะคืนค่ารีจิสเตอร์ที่เก็บไว้คืนให้ด้วย ซึ่งถ้าท่านผู้อ่านท่านใดยังสงสัยก็ขอให้ย้อนกลับไปดูรายละเอียดในตอนที่ผ่านมา

การเปลี่ยนค่าในตารางอินเตอร์รัพต์เวกเตอร์

วิธีการเปลี่ยนค่าในตารางอินเตอร์รัพต์สามารถทำได้โดยการเขียนตำแหน่งหน่วยความจำของโปรแกรมย่อยของเราลงไปในช่องตารางอินเตอร์รัพต์ที่ต้องการ ซึ่งแต่ละช่องของตารางจะเก็บพอยน์เตอร์แบบไกล (Far pointer) โดยประกอบไปด้วยสองส่วนคือ โค้ดเซกเมนต์ (Code Segment) และออฟเซต (offset) ของตำแหน่งโปรแกรมนั่นเอง อย่างไรก็ตามในเทอร์โบซีเองก็ได้เตรียมฟังก์ชันชื่อ setvect ที่ทำหน้าที่รับพารามิเตอร์คือพอยน์เตอร์แบบไกล และหมายเลขอินเตอร์รัพต์ที่ต้องการเปลี่ยน ตัวอย่างเช่น ต้องการนำเอาฟังก์ชัน mykeyboard ไปแทนในตารางช่องที่ 0X16 ก็จะทำได้ดังในรูปที่ 2

การเชื่อมโยงกับโปรแกรมบริการอินเตอร์รัพต์เดิม

ในการใช้งานอินเตอร์รัพต์ หมายเลขที่ถูกใช้งานโดยระบบหรือโปรแกรมจะต้องมีความระมัดระวังให้ระบบหรือโปรแกรมอื่นทำงานได้อย่างถูกต้องตามปกติ ซึ่งก็มักจะทำให้เกิดการทำงานของโปรแกรมบริการอันเดิมภายในโปรแกรมบริการที่เราสร้างขึ้นมาใหม่ด้วย และการทำงานของโปรแกรมบริการเดิมนั้นก็ทำได้โดยเรียกตำแหน่งของโปรแกรมบริการเดิมนั้นเอง ซึ่งการที่จะเรียกเอาโปรแกรมบริการเดิมขึ้นมา

```
void interrupt mykeyboard (IREGS iregs)
 .                .
 .                .
 .                .
Setvect (0X16, mykeyboard);
 .                .
 .                .
```

รูปที่ 2 แสดงการเปลี่ยนตารางอินเตอร์รัพต์เวกเตอร์ โดยใช้ฟังก์ชัน setvect ซึ่งหลังจากเปลี่ยนค่าในตารางโดยนำเอาตำแหน่งหรือพอยน์เตอร์ของฟังก์ชัน mykeyboard ไปใส่แทนแล้วเกิดอินเตอร์รัพต์ 0X16 ขึ้น ฟังก์ชัน mykeyboard ก็จะถูกเรียกใช้งานแทนทุกครั้ง

```
void interrupt (*oldkeyboard) (void);
      .              .
      .              .
      .              .
oldkeyboard = getvect (0X16)
      .              .
      .              .
      .              .
```

รูปที่ 3 การเก็บเอาตำแหน่งโปรแกรมบริการอินเตอร์รัพต์เดิมเอาไว้ในตัวแปรเพื่อใช้เรียกในฟังก์ชันบริการอินเตอร์รัพต์ที่เราสร้างขึ้นมาเอง

ทำงานก็จะทำได้โดยต้องเก็บเอาตำแหน่งโปรแกรมบริการเดิมเอาไว้ก่อนด้วยฟังก์ชันพิเศษที่เทอร์โบซีเตรียมไว้ให้คือฟังก์ชัน getvect ดังในรูปที่ 3

โปรแกรมจับเวลาสารพัดประโยชน์

เอาล่ะครับหลังจากเรียนรู้ทฤษฎีและวิธีการใช้งานแล้ว ก็ต้องตามด้วยตัวอย่างเสริมอีกสักตัวอย่างหนึ่ง ซึ่งตัวอย่างที่นำมาเสนอผู้อ่านสามารถนำไปใช้ประโยชน์ได้ด้วย โดยเน้นการนำเอาอินเตอร์รัพต์ของเวลามาใช้เพื่อการจับเวลา ซึ่งในหลาย ๆ งานต้องใช้ช่วงเวลาสำหรับทำงานที่กำหนดโดยแม่นยำ ตัวอย่างเช่น การตรวจสอบสถานะของอุปกรณ์ที่ต่อพ่วงอยู่กับระบบโดยต้องทำการตรวจสอบทุก ๆ ช่วงเวลาแล้วนำมาแสดงผล หรือใช้ในการรองรับอินพุตข้อมูลจากผู้ใช้โปรแกรมซึ่งจะรอผู้ใช้ในช่วงเวลาที่กำหนดเช่น 30 วินาที การทำงานลักษณะนี้สามารถนำเอาโปรแกรมในรูปที่ 4, 5 และ 6 ไปช่วยใช้งานได้และในรูปที่ 5 ไฟล์ชื่อ Timer.h ซึ่งเป็นหัวไฟล์ของไฟล์โปรแกรมในรูปที่ 6 ซึ่งเป็นตัวฟังก์ชัน การนำไปใช้งานในโปรแกรมนั้นก่อนอื่นใดจะต้องเรียกฟังก์ชันชื่อ intercept_timer ซึ่งในการทำงานของฟังก์ชันนี้จะไปเปลี่ยนค่าในตารางอินเตอร์รัพต์เวกเตอร์ของอินเตอร์รัพต์เวลาให้ชี้ไปยังฟังก์ชันชื่อ newtimer ซึ่งจะทำหน้าที่ในการนับเวลา ถ้าหากเวลาที่ตั้งเอาไว้หมดมันจะไปเรียกฟังก์ชันที่เก็บอยู่ในตัวแปรชื่อ timeout_function โดยค่าในตัวแปร timeout_function ถูกกำหนดโดยการผ่านพารามิเตอร์ให้กับฟังก์ชัน intercept_timer อย่างไรก็ตาม ถ้าไม่ต้องการให้ทำงานอะไร เมื่อเวลาหมดก็ให้ใส่พารามิเตอร์ในฟังก์ชัน intercept_timer เป็นค่า NULL แทน หลังจากเรียกฟังก์ชัน intercept_timer เพื่อติดตั้งฟังก์ชันบริการอินเตอร์รัพต์เรียบร้อยแล้วการจะใช้งานก็จะต้องตั้ง

```
/* ------------ interupt.h ------------ */

/* ------- the interrupt function registers -------- */
typedef struct {
   int bp,di,si,ds,es,dx,cx,bx,ax,ip,cs,fl;
} IREGS;

#define VIDEO 0x10
#define DOS   0x21
#define KYBRD 9
#define KEYBOARD 0x16
#define TIMER 8

#define KBDATA     0x60
#define KBCONTROL  0x61
#define KBSTATUS   0x64

#define ZEROFLAG 0x40
```

รูปที่ 4 ไฟล์ interupt.h ซึ่งเป็นไฟล์ที่ใช้ประกาศค่าคงที่ของหมายเลขอินเตอร์รัพต์ที่ต้องการใช้งาน

```
/* ---------- timer.h --------------- */
extern unsigned ticker;

/* -------- timer prototypes --------- */
void intercept_timer(void (*)(void));
void restore_timer(void);

/* ------------ timer macros ------------- */
#define timed_out() (ticker==0)
#define set_timer(secs) ticker=(secs)*182/10+1
```

รูปที่ 5 ไฟล์ timer.h ซึ่งไฟล์สำคัญประกาศรูปแบบและมาโครฟังก์ชันของโปรแกรมจับเวลาอเนกประสงค์

```
/* ---------- timer.c --------------- */
#include <stdio.h>
#include <dos.h>
#include "timer.h"
#include "interupt.h"

unsigned ticker;

static void (interrupt far *oldtimer)(void);
static void interrupt far newtimer(void);
static void (*timeout_function)(void);

/* ---- intercept the timer interrupt vector ----- */
void intercept_timer(void (*fn)(void))
{
    if (oldtimer == NULL)    {
        oldtimer = getvect(TIMER);
        setvect(TIMER, newtimer);
    }
   timeout_function = fn;
}

/* ---- restore timer interrupt vector ------- */
void restore_timer()
{
    if (oldtimer != NULL)
        setvect(TIMER, oldtimer);
   timeout_function = NULL;
}

/* ------ ISR to count timer ticks ------- */
static void interrupt far newtimer()
{
    (*oldtimer)();
    if (ticker > 0)
        if (--ticker == 0 && timeout_function != NULL)
         (*timeout_function)();
```

รูปที่ 6 ไฟล์ timer.c เป็นไฟล์ที่เก็บฟังก์ชันต่าง ๆ สำหรับจับเวลาอเนกประสงค์

จำนวนช่วงเวลาที่จะให้นับถอยหลังโดยเรียกมาโครฟังก์ชัน set_timer ที่กำหนดเอาไว้ในไฟล์ timer.h ซึ่งช่วงเวลาที่ใช้กำหนดจะมีหน่วยเป็นวินาที ถ้าสังเกตในมาโคร set_timer จะนำเอาจำนวนวินาทีคูณกับ 18.2 แล้วเก็บผลลัพธ์เอาไว้ในตัวแปร ticker ซึ่งถูกใช้ในการนับค่าลงทีละหนึ่ง แต่ละครั้งที่อินเตอร์รัพต์จากเวลาของระบบเกิดขึ้น จนกระทั่งค่าในตัวแปร ticker มีค่าเป็นศูนย์ นั่นหมายถึงช่วงเวลาที่ตั้งเอาไว้หมด และสามารถจะตรวจพบว่าเวลาหมดหรือยังโดยใช้มาโครฟังก์ชันชื่อ timed_out ซึ่งกำหนดเอาไว้ในไฟล์ timer.h เหมือนกัน หรืออาจจะไม่ต้องทำการตรวจสอบเลยก็ได้ ถ้าเราได้ติดตั้งฟังก์ชันที่จะให้เริ่มการทำงานเมื่อเวลาหมด (ติดตั้งเอาไว้ตอนเรียกฟังก์ชัน intercept_timer) ดังนั้นในกรณีนี้เราสามารถทำงานอื่นใดก็ได้ หลังจากได้ตั้งเวลาแล้ว ซึ่งเมื่อการจับเวลาสิ้นสุดลงแล้วก็จะเกิดการทำงานในฟังก์ชันนั้นเองโดยอัตโนมัติ จะเห็นว่าการนับเวลาจะไม่กระเทือนผลการทำงานใด ๆ กับโปรแกรมหลักของเรา และเราเองก็สามารถจะกำหนดช่วงเวลาสำหรับนับใหม่ได้ ด้วยฟังก์ชัน set_timer ตามความต้องการ สิ่งที่สำคัญและลืมไม่ได้ก็คือ ก่อนจะจบโปรแกรมหรือเลิกการทำงานจะต้องทำการเรียกฟังก์ชัน restore_timer ด้วยเสมอ เพื่อที่จะได้คืนเอาค่าเดิมลงในตารางอินเตอร์รัพต์เวกเตอร์ เพราะถ้าไม่นำค่าเดิมคืนหลังจากเลิกโปรแกรมแล้ว หน่วยความจำที่เคยเป็นของโปรแกรมซึ่งตอนนี้เป็นที่ว่างไปแล้ว เมื่อเกิดการอินเตอร์รัพต์อีกก็อาจจะทำให้ระบบหยุดทำงานหรือเสียหายได้

สรุป

ในมุมภาษาซีฉบับนี้ ได้เสริมความรู้ในเรื่องของโครงสร้างการอินเตอร์รัพต์ในระบบ พร้อมการใช้งานและตัวอย่าง ซึ่งคิดว่าท่านผู้อ่านคงได้นำไปใช้ประโยชน์ได้บ้าง เพราะการพัฒนาโปรแกรม หลายงานต้องเข้าไปยุ่งเกี่ยวกับระบบโดยตรงอย่างหลีกเลี่ยงไม่ได้ ในตอนต่อไปผู้เขียนจะได้นำเอาตัวอย่างมาเสริมให้ท่านอ่านอีกตัวอย่าง ซึ่งจะเป็นเรื่องของการสร้างโปรแกรมเรสซิเดนต์ ที่ช่วยรักษาจอภาพโดยใช้เรื่องของการอินเตอร์รัพต์เข้ามาช่วย รวมทั้งโปรแกรมจับเวลาอเนกประสงค์ในฉบับนี้ด้วย พบกันฉบับหน้าครับ